

Dynamic Cloth Manipulation with Deep Reinforcement Learning

Rishabh Jangir, Guillem Alenyà, Carme Torras

Abstract—In this paper we present a Deep Reinforcement Learning approach to solve dynamic cloth manipulation tasks. Differing from the case of rigid objects, we stress that the followed trajectory (including speed and acceleration) has a decisive influence on the final state of cloth, which can greatly vary even if the positions reached by the grasped points are the same. We explore how goal positions for non-grasped points can be attained through learning adequate trajectories for the grasped points. Our approach uses few demonstrations to improve control policy learning, and a sparse reward approach to avoid engineering complex reward functions. Since perception of textiles is challenging, we also study different state representations to assess the minimum observation space required for learning to succeed. Finally, we compare different combinations of control policy encodings, demonstrations, and sparse reward learning techniques, and show that our proposed approach can learn dynamic cloth manipulation in an efficient way, i.e., using a reduced observation space, a few demonstrations, and a sparse reward.

I. INTRODUCTION

Day to day tasks of a household robotic assistant would involve manipulating deformable objects (cloth folding [1], bed making [2], getting dressed [3], [4]). But the majority of state-of-the-art robotic manipulation work focuses on rigid objects. Progress towards deformable object manipulation has been scarce, because manipulating deformable objects poses additional challenges over rigid objects. The shape of deformables varies largely along and between trajectories with the same end points and it is difficult to characterize due to their high-dimensional configuration spaces.

Deformable object manipulation can be achieved by static or dynamic manipulations. In dynamic manipulation, forces due to acceleration play a relevant role along with kinematics, static and quasi-static forces. Dynamic movement permits controlling non-grasped points as well, increasing in a way the dexterity of manipulation process at the expense of the increased complexity of underactuation. Recent research in deformable object manipulation ([5], [6], [7]) mostly considers only static manipulation tasks. The majority focus on explicit physics-based modeling of deformable behavior in simulation and they attempt to find an optimal trajectory to guarantee the desired outcome. Some studies have tried dynamic manipulations of flexible objects ([8], [9], [10]) such as (dynamic) cloth folding, rope knotting, etc. with

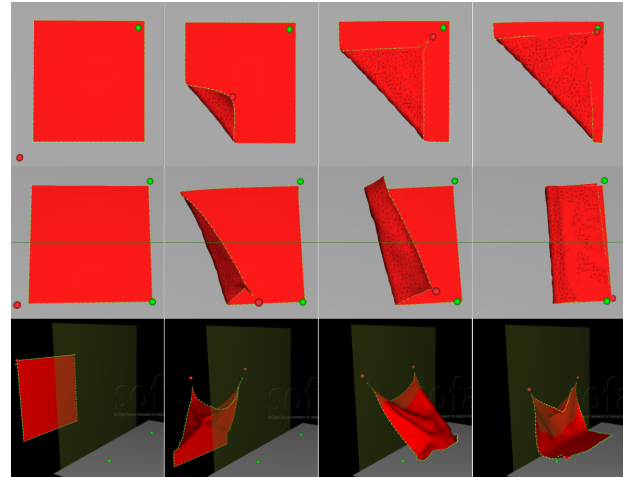


Fig. 1: Frames from rollouts of the learned policy are shown above for three tasks. The agent (red sphere) interacts with the cloth to move selected vertex/s to their target positions. These are complex, under-actuated dynamic tasks where, to succeed, the agent needs to learn to successfully manipulate parts of cloth which are not in its direct control.

physics-based modeling ([11]). A downside of these approaches and some other approaches that rely on visuomotor servoing ([12], [13], [14]) is the requirement of significant engineering specific to the manipulation task. Additionally, even while only considering static manipulation these models tend to be very sensitive to the deformation parameters of the object. It is then safe to say that these approaches would fail when considering dynamic manipulation tasks where the task performance heavily relies on the deformation parameters.

Alternative approaches that employ trial-and-error-based machine learning methods ([15], [16], [17], [18], [19], [20]) to map observations directly to actions have shown interesting results on robotic manipulation of deformable objects, but they restrict themselves to static manipulation as well. Regarding dynamic manipulation, [21] developed a control method to realize a target state by calculating an optimized time-series joint torque command, but they mainly consider only a 2D actuation that allows only simple manipulation tasks and [22] considers only quasi-static movements.

Currently, training these model-free learning-based approaches is highly sample inefficient. Fortunately, simulations provide an efficient way to train control policies. However, accurate simulation of deformable objects is challenging. Previous approaches have investigated the usage of several simulators for deformable objects ([23], [24]) but none of them have proved to be a benchmark for textile

*This work has been supported by the ERC project Clothilde (ERC-2016-ADG-741930), the HuMoUR project TIN2017-90086-R (AEI/FEDER, UE) and by the AEI through the María de Maeztu Seal of Excellence to IRI (MDM-2016-0656).

The authors are with Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Llorens i Artigas 4-6, 08028 Barcelona, Spain. {rjangir, galenya, torras}@iri.upc.edu

object simulation (like Mujoco [25] for rigid objects).

Large configuration spaces can possibly be tackled by using RGB observation data. However, this would increase the computing cost by a large amount and, in addition, it is challenging to learn directly in the high-dimensional input state-space, which can also be redundant given the aim of the task. Moreover, current learning algorithms that use RGB information to solve cloth manipulation tasks [18] note the need of auxiliary inputs for the learning to succeed anyway. Nevertheless, they restrict themselves to solving static manipulations tasks only.

In order to address these problems, in this work we employ a model-free deep reinforcement learning (RL) method to learn dynamic manipulation of textile objects. Our method involves minimal task engineering as we take advantage of very few demonstrations and sparse rewards. This has been extensively studied in the context of rigid object manipulation [26], [27], [28], [29], and a few studies on static deformable object manipulation [18], [30], but to the best of our knowledge no study has previously investigated the applicability of deep RL methods to *dynamic* deformable object manipulation tasks. We use the SOFA simulator [31] to define 3 textile manipulation tasks (Fig. 1): one static manipulation task of folding a napkin diagonally, and two dynamic manipulation tasks of folding a napkin sideways and placing a napkin on a table. We provide a single sparse reward on task completion in all the tasks. Due to the challenging nature of the dynamic tasks being considered, we omit RGB information and directly work on the simplified clothing state consisting of a few distinguished points.

Through our experiments, we (a) demonstrate the basic difference between static and dynamic manipulations and show that the simulator is capable of capturing these differences. (b) Investigate the trade-off between complexity and effectiveness to find an accurate textile state representation for our tasks. (c) Show that our method can learn valid dynamic manipulation behaviors using a low-fidelity simulation platform and test different combinations of the components of the algorithm to show their individual effect.

II. BACKGROUND

We consider a standard RL problem where an agent interacts with an environment according to a policy in order to maximize rewards over discrete timesteps. The framework considers partially observable environments that are modeled as continuous state partially observable Markov decision processes (POMDPs) defined as a tuple $(S, A, P, r, O, \rho_0, \gamma)$, where S is a set of full states of the environment, A is a set of continuous actions, $P : S \times A \times S \rightarrow R$ is the transition probability distribution, $r : S \times A \rightarrow R$ is the reward function, ρ_0 is the initial state distribution, and $\gamma \in (0, 1]$ is the discount factor. The decision process is partially observable and the agent receives observations o from the set of observations O .

The goal of the agent is to maximize the multi-step return $R_t = \sum_{t'=t}^T \gamma^{t'-t} r_{t'}$, where T is the fixed horizon of each episode. The objective during learning is to find an optimal

policy $\pi^* : O \rightarrow A$ that maximizes the expected return of the agent $J(\pi)$

$$\pi^* = \underset{\pi}{\operatorname{argmax}} J(\pi). \quad (1)$$

Alternatively, the expected return upon taking an action a_t in state s_t can be measured by a Q function $Q(s_t, a_t) = E[R_t | s_t, a_t]$. In terms of Q function the objective can be written as,

$$\pi^* = \underset{\pi}{\operatorname{argmax}} Q(s_t, \pi(o_t)). \quad (2)$$

The transition probability distribution P determines the consequences of the agent's actions and is dependent on the dynamics of the environment. The dynamics is therefore of crucial importance, as it determines the behaviors that can be realized. Deformable objects introduce an additional component in the environment dynamics which is lacking for rigid objects, making it more difficult to learn valid behaviors without direct access to the environment dynamics.

A. Deep Deterministic Policy Gradients

DDPG [32] is a policy gradient algorithm for learning control policies in continuous action domain. It uses off-policy data and the Bellman equation to concurrently learn the Q -function and a policy. It uses an actor neural network (policy network), parameterized by a set of parameters θ^π , that maps observations to actions $\pi(\theta) : O \rightarrow A$ and tries to maximize $Q(s_t, \pi_\theta(o_t))$ at each time-step t . However, the Q function is not known and DDPG employs a critic neural network, parameterized by parameters θ^Q , to estimate Q values of actions at each time-step t .

During training, the agent acts in the environment according to noisy policy $a_t = \pi(o_t) + N(0, \sigma)$. The Gaussian noise facilitates exploration. Each transition the agent generates is stored in a replay buffer R from where it is sampled in batches of N tuples to train the networks. Sampling from a replay buffer stabilizes training by removing temporal correlations and therefore reduces the changes in the distributions the networks are trying to learn. DDPG also employs target networks Q^* and π^* to reduce the risk of Q value estimates oscillating or diverging due to the recursive Q value definition in the Bellman equation.

B. Universal Policy

A universal policy [33] is a simple extension where the goal $g \in G$ is provided as an additional input to the policy $\pi(a|o, g)$. The reward is then also dispensed according to the goal $r(s_t, a_t, g)$. In our framework, a different goal will be randomly sampled at the start of each episode, and held fixed over the course of the episode. For the cloth manipulation tasks, the goal specifies target location for the selected vertices as discussed later.

C. Hindsight Experience Replay

Learning from a sparse binary reward is known to be challenging for most modern RL algorithms. We will therefore leverage a recent innovation, Hindsight Experience Replay (HER) [34], to train policies using sparse rewards. In this work, we consider sparse binary rewards of the

form, $r(s, g) = 0$ if g is satisfied in s , and $r(s, g) = -1$ otherwise. Consider an episode with trajectory $\tau \in (s_0, a_0, \dots, a_{T-1}, s_T)$, where the goal g was not satisfied over the course of the trajectory. Since the goal was not satisfied, the reward will be -1 at every timestep, providing little information to the agent.

But suppose that we are provided with a mapping $m : S \rightarrow G$, that maps a state to the corresponding goal satisfied in the given state. For example, $m(s_T) = g'$ represents the goal that is satisfied in the final state of the trajectory. Rewards are then recomputed under the new goal g' . The trajectory will be successful under the new goal as g' is satisfied in s . By replaying past experiences with HER, the agent can be trained with more successful examples than those available in the original recorded trajectories. We have considered replaying 4 goals from the future states of a trajectory, but HER is also amenable to other replay strategies.

D. Learning from demonstrations

Exploration in continuous state-space environments is another existing challenge for deep RL algorithms. Moreover, the exploration problem is magnified when we consider deformable object environments due to their large configuration spaces. We combine DDPG algorithm with several practical extensions as introduced in ([27], [35]), namely an additional buffer to store demonstration data, a modified loss that uses behavior cloning loss as an auxiliary loss, and a filter on the Q values which accounts for sub-optimal demonstrations.

III. METHOD

A. Simulation

Development towards finding effective deformable object simulators has been scarce. This is because of the difficulty of efficiently simulating deformations and the large diversity observed in case of deformable objects. Although there have been efforts towards using existing simulators that support deformable objects for deep RL research ([23], [24]) there have been no benchmark simulations or tasks. An important case of deformable objects is textile/clothing. Matas et al.[18] used PyBullet [23] for simulating towel manipulation tasks, but they reported issues with the simulator codebase such as instability as it only implements some rudimentary and experimental functionality. Instead, for this work we use the SOFA simulation framework [31]. SOFA provides us with more realistic and intricate deformations of the clothing. In comparison with PyBullet we found that SOFA was much more stable and flexible. Although the simulator still does not support self-collisions in the cloth, it provides functionalities to define the physical properties associated with the kind of deformation required.

B. Cloth manipulation environments

We designed and implemented 3 environments using OpenAI gym [36] API for solving manipulation tasks involving a square cloth of fixed shape and size placed on or around a rigid table. The cloth is modeled as a mesh of triangles joined together by their vertices. We call these vertices nodes in the

cloth mesh, mass of the cloth is uniformly spread among all nodes. A rigid ball is used as a manipulator which can attach itself to a node on the cloth. The central aim of this work is to study dynamic manipulation of textiles, thus we bypass the need to model a physical grasp in the simulation. Instead, we use a fake grasp implemented as a binary point grasp to manipulate the textile. Creation of the grasp constraint is subject to the manipulator being in close proximity to a cloth node. Other forces acting on the clothing apart from the manipulator are gravity and interaction forces with the table such as friction.

C. Tasks

Fig. 2 shows the textile manipulation tasks we consider in the environments defined above. Each task runs for a fixed number of simulation steps T (different for different tasks) before it automatically gets reset to the initial state. At the start of each episode, the manipulator is initialized to a default pose near the vertex to be manipulated, and the cloth is placed randomly within a fixed bound on the table or around it depending on the task. There are no deformations/wrinkles in the cloth at the start of an episode. Goal state g varies for each task depending on the aim and definition of tasks which we discuss later. For dynamic manipulation tasks the motion of manipulator is constrained within a predefined workspace in order to demonstrate the emergence of dynamic behaviors. The 3 environments are,

Diagonal (Static) Folding: This task involves diagonally folding a piece of cloth lying on a table. Goal state is achieved if the vertex being manipulated is within a threshold distance ($\delta=10$ units) of the goal location. The goal location is randomly sampled along the diagonal in close proximity to the opposite vertex. No restrictions on the workspace of the manipulator in this task.

Sideways (Dynamic) Folding: This task involves sideways folding a piece of cloth lying on a table. Goal state is achieved if both pairs of adjacent vertices are within a threshold distance ($\delta=10$ units) of goal location. The goal locations are randomly sampled close to the destination vertices. The manipulator workspace is constrained in such a way that it cannot reach the other half of the cloth. The only way to manipulate the vertex in the other half of the cloth then is to rely on the fabric connection between the vertices and swing the other vertex to its goal location.

(Dynamic) Placing on a table: This task involves partially placing a piece of cloth on a table. The cloth hangs parallel to an edge of the table outside the table space. Two vertices of the cloth are grasped by two rigid ball manipulators at all times and the other two vertices are freely hanging. Goal state is achieved when both hanging vertices are within a threshold distance ($\delta=20$ units) of goal locations. Two goals are randomly sampled such that the line connecting them is parallel to the edge of the table. Similar actions are applied to both the manipulator and action space for this task is 3D. The manipulator workspace is constrained and thus it cannot go beyond the edge of the table. The only way to reach the

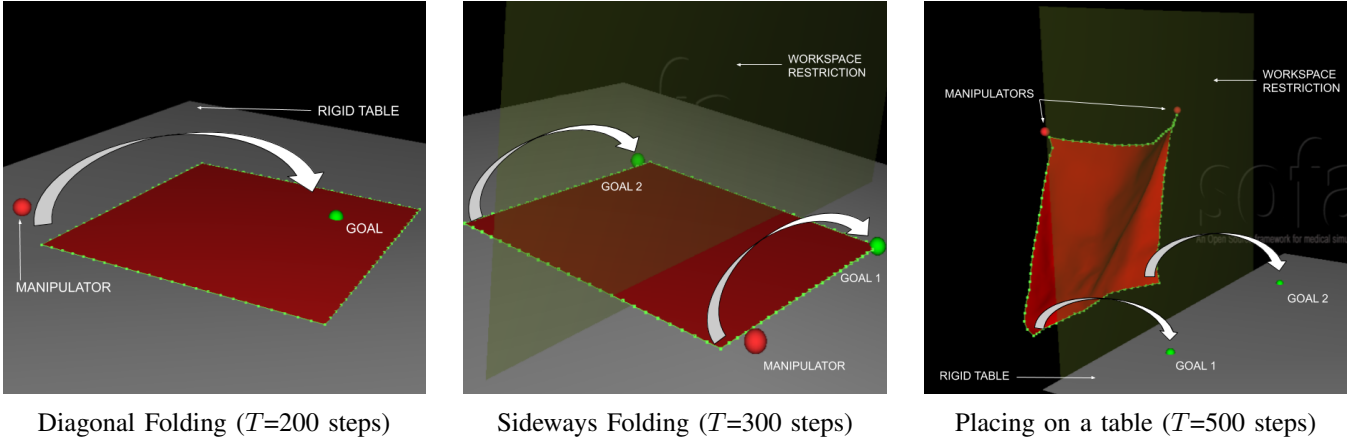


Fig. 2: Task definition diagrams are shown for each task. The goal state is reached when selected vertice/s are within a certain threshold of their target positions. The target spheres (green) do not interact with the environment. In the last two tasks the agent’s workspace is restricted as shown by a green plane and thus dynamic manipulations are required to successfully reach the goal state in these tasks.

goal state is to swing the cloth such that the hanging vertices land on the goal locations.

D. State and Action

The state is represented using positions and velocities of selected nodes on the cloth, position and velocity of the manipulator, position of the goal/s, as well as grasp state of the manipulator. The combined features result in a state space ranging from 34D to 85D depending on the number of nodes selected on the cloth and the task. Actions from the policy specify target velocities for the manipulator in x, y and z directions as well as a boolean gripping action. Manipulator rotation is not necessary for the tasks and is therefore kept fixed. This yields a 4D action space for the first two tasks and 3D action space for the last task where gripping action is not used.

E. Learning behaviors from demonstrations algorithm

During initial exploration we found that DDPG alone was not successful in solving any of the proposed tasks. We suspected the failure was due to the additional complexity that deformable objects bring to the environment with their infinite configuration spaces. So we investigated possible improvements ([27], [35]) alongside vanilla DDPG. This led us towards employing demonstrations to aid the agent in faster exploration. Firstly, an additional buffer R_D was initialized to contain demonstration data, at training time we draw an extra N_D samples from this demonstration buffer along with environment interaction data from main buffer in batches. Secondly, behavior cloning loss L_{BC} was used as an auxiliary loss to train the actor network only on the demonstration samples in the data defined as,

$$L_{BC} = \sum_{i=1}^{N_D} (\pi(o_i, Q_\pi) - a_i)^2 \mathbf{1}_{Q(s_i, a_i) > Q(s_i, \pi(o_i))}. \quad (3)$$

In DDPG, a critic network is trained to predict Q values of the actions taken by the actor. The final part in Equation 3

corresponds to a filter on the Q value update, that allows update only if the demonstrated action had a better Q value than the action output by the policy network. This results in the following loss functions, L_π for the actor

$$L_\pi = \lambda_1 \nabla_{\theta_\pi} J - \lambda_2 \nabla_{\theta_\pi} L_{BC} \quad (4)$$

where λ_1 and λ_2 are hyper-parameters corresponding to the weight shared between the two loss terms in L_π , and L_Q for critic network,

$$L_Q = \frac{1}{N} \sum_i ((r_i + \gamma Q(s_{i+1}, \pi(o_{i+1})) - Q(s_i, a_i | \theta_Q))^2. \quad (5)$$

Training the agent with demonstrations aided in exploration and it was able to solve the Diagonally folding task, still it failed in the two tasks involving dynamic manipulation. This can be attributed to the complex nature of the dynamic tasks and only sparse rewards being provided upon success. We further employed HER to handle the sparse nature of our data. HER augments the original training data with more successful examples by replaying trajectories with modified goals. HER only works with stationary goals and thus we designed tasks in such a way that the goal locations are fixed throughout the episode. We study the performance of individual components of the algorithm in Section IV.

F. Capturing dynamics in a simulation

The basic difference in dynamic manipulation as compared to static one is the manipulator being able to control non-grasped vertices. For example, in the Sideways Folding task the manipulator relies on the connection between grasped vertex (no.3) and the vertex to be manipulated (no.2) to successfully achieve the goal state. Effectively, the manipulator is being able to control parts of the object outside its workspace. The manipulating agent’s ability should thus encompass learning deformation behavior as well as the effects of manipulator’s velocity and path on the outcome of the task. Consequently, the simulator must be able to

Randomization	Diagonal Folding (Static)	Sideways Folding (Dynamic)	Placing on table (Dynamic)
None	1.0 ± 0.00	0.99 ± 0.01	0.52 ± 0.04
Speed	0.99 ± 0.01	0.41 ± 0.07	0.41 ± 0.06
Trajectory	1.0 ± 0.00	0.12 ± 0.03	0.09 ± 0.05
Speed+Trajectory	1.0 ± 0.0	0.08 ± 0.04	0.14 ± 0.02

TABLE I: Success rate comparisons for all three tasks under different randomization. Note that randomization has negligible effect on success rate for static tasks.

capture these differences in our chosen tasks. In Section IV we present experimental results to show the relevance of manipulator trajectory, velocity and acceleration in the case of dynamic manipulation tasks and the simulator’s ability to differentiate between task-relevant motion features.

G. Generating demonstrations

The demonstrations for all the tasks are generated by a hard-coded python script. The actuator follows points in the cloth reference frame that are pre-selected by a human in just one instance of the tasks. A uniform Gaussian noise of 10% is added to introduce randomness in the demonstration data. The generated demonstrations are imperfect and we use 20 episodes of demonstration data to train our agent.

IV. EXPERIMENTS

We have performed three sets of experiments to study the effectiveness of reinforcement learning in solving dynamic deformable object manipulation tasks. The first set aims to unravel the basic difference between dynamic and static manipulation. The second is intended to assess the influence of state representation on performance. And the third tries to solve a difficult dynamic manipulation problem on deformable environments that had not yet been studied in the RL community, also we study the effect of individual components of the algorithm.

A. Capturing dynamics in a simulation

In order to assess the importance of task-relevant motion features, we introduce randomization in the speed and the trajectory of manipulator while keeping initial and final positions intact in our hard-coded demonstration generation script. In Table I we report success rates under different randomization for all three tasks averaged over 3 epochs of 100 episodes each.

Success rates for dynamic tasks drop significantly under all types of randomization whereas negligible effect on the success rate of the static task is observed. This can be attributed to the fact that in the Diagonal Folding task the vertex in consideration is always grasped by the manipulator throughout the trajectory, so only final position of the manipulator matters for task success. Thus we can conclude that, in dynamic tasks speed and trajectory play an important role in deciding task success. Additionally the results suggest that the simulator is capable of capturing the dynamic information in the proposed tasks.

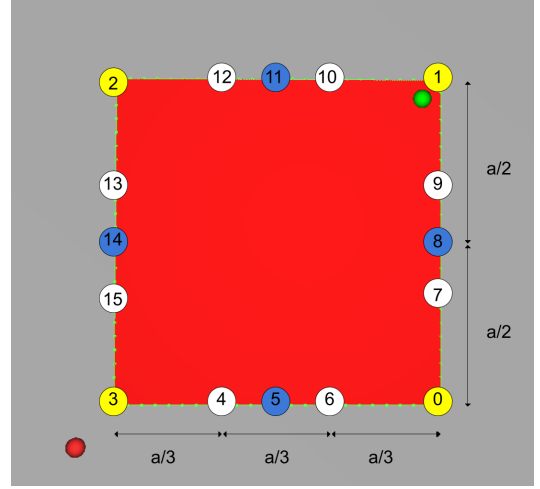


Fig. 3: Positions of selected points on the cloth. The 3 input observation spaces considered are: (1) 4 yellow points, (2) 4 yellow and 4 blue points, (3) 4 yellow and 8 white points.

B. Textile state representation

To investigate the effectiveness of different textile state representations, we compare agents trained with 3 different configurations of observation spaces as defined in Fig. 3. The main difference between the three alternatives is the size of observation space which increases from 4 to 12. We train our best performing "DDPG+Demo+HER" agent (as defined in Sec. IV-C) with 3 random seeds for each configuration. Learning curves comparing the performance of each input space can be seen in Fig. 4.

We observe that 4 points is an insufficient state representation for solving the proposed tasks. A sudden drop in performance can be observed for this case which is possibly due to the algorithm learning a sub-optimal policy given how little deformation information is captured by only 4 vertex points. All the tasks show an improvement in performance (higher success rate, stable learning curves) as we increase the observation space to 8 points. Further increment in the observation space size does not affect the performance in Diagonal Folding and Sideways Folding tasks as can be seen by the overlapping orange and green curves. Whereas for the same change, decrement in performance for Placing on table task is observed. Note that, increasing the observation space size increases training time of the algorithm. Therefore we can conclude 8 points are the best choice of observation space for the tasks considered in this work.

C. Dynamic manipulation of deformable objects

We train 3 agents on our proposed tasks in simulation and do performance comparisons to study the importance of individual components. All 3 agents use DDPG as their base algorithm with 4 layered actor and critic networks. Also, they share the same configuration of all hyperparameters except the replay buffer data which also includes demonstration data ($N_D = 1/8 \times \text{Total sampled data}$) and modified actor loss L_π to $\lambda_1 = 0.001, \lambda_2 = 0.0078$ from $\lambda_1 = 1.0, \lambda_2 = 0.0$ in the case of learning from demonstrations. We consider HER and

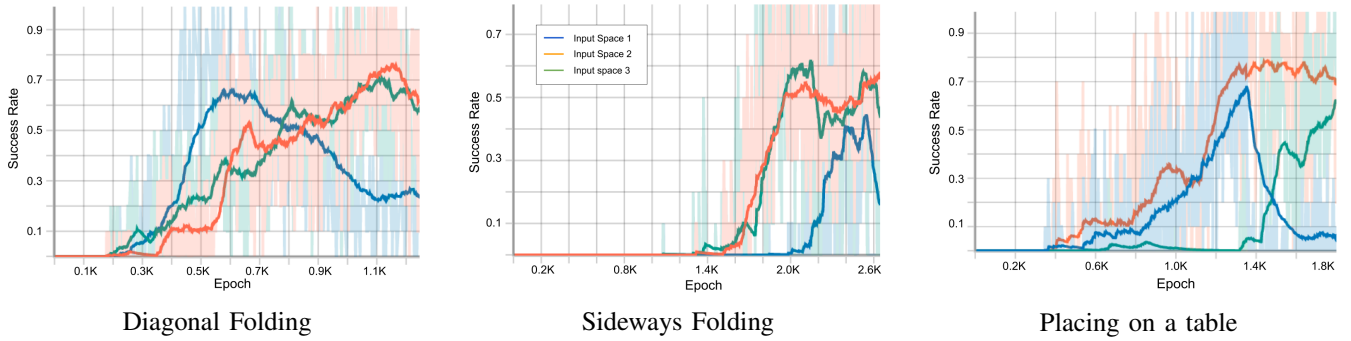


Fig. 4: Learning curves comparing performance of different observation space inputs. We use 3 random seeds per method and report average success rate. The median of the runs is shown in bold and each training run is plotted in a lighter color.

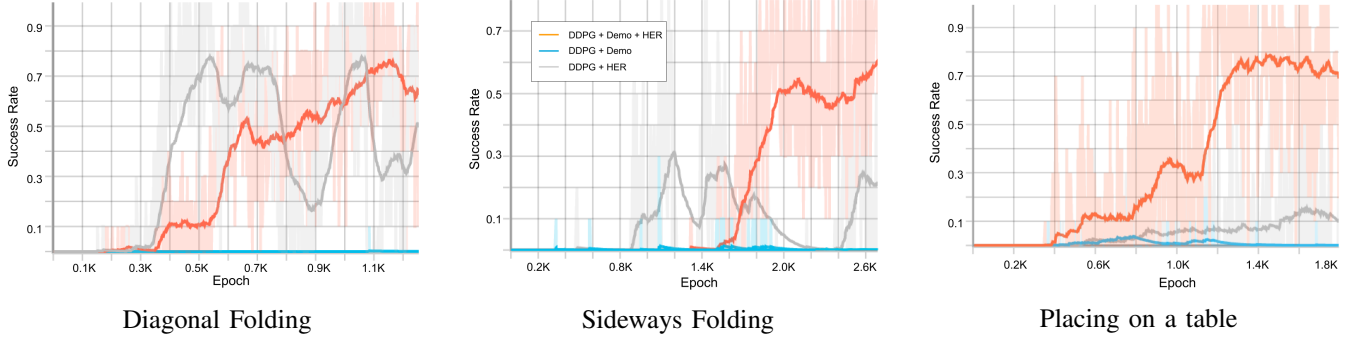


Fig. 5: Learning curves comparing training with and without demonstrations and HER along with DDPG. We use 3 random seeds per method and report average success rate. Note that the "DDPG+Demo" method without HER is always at around 0% success rate. As the difficulty of the task increases demonstrations become essential for good performance.

training with demonstrations (Demo) as improvements and thus the 3 agent combinations are "DDPG+Demo+HER", "DDPG+HER" and "DDPG+Demo". We use 8 points for input space as determined in the previous experiment. At training time, each epoch consists of 20 episodes of training followed by 10 episodes of policy testing. The success rate at each epoch is measured as the average over 10 test episodes. We train each agent with 3 random seeds. Qualitative performance of the learned policies for the three tasks can be seen in the attached video where different executions of the learned policy are demonstrated and in Fig. 1, whereas learning curves comparing the performance of each agent can be seen in Fig. 5.

The "DDPG+Demo+HER" agent is able to solve all of the proposed tasks and successfully learn valid dynamic manipulation behaviors for the Sideways Folding and Placing on a table tasks. Looking at the learning curves, we observe that training without HER is unable to solve any of the tasks. Moreover, training without demonstrations is able to achieve comparable success only for the static task but the learning is highly unstable. This can be attributed to insufficient exploration and reaching sub-optimal minima without the presence of demonstrations to guide the agent towards good behaviors. An example of a sub-optimal behavior we observed while training for the diagonal folding task is the agent learning to grasp the cloth around midpoint of the line joining goal

position and vertex and trying to swing the vertex to goal. The importance of demonstrations is magnified in the case of dynamic tasks where performance of "DDPG+HER" agent is significantly lower. But demonstrations are of little use for the above tasks when training without HER as observed with the "DDPG+Demo" agent.

V. CONCLUSIONS

Building upon recent work in end-to-end learning for rigid object manipulation, we extended the ideas to dynamic cloth manipulation. We demonstrated the importance of speed and trajectory in the case of dynamic manipulations and investigated the effectiveness of different textile state representations. By restricting the manipulator workspace we showed the emergence of dynamic manipulation behaviors and successfully solved 3 long horizon tasks: Diagonal Folding, Sideways Folding and Placing on a table, using a deep RL method which bypasses the need to explicitly model cloth behavior and does not require reward shaping to converge. Also, we studied the importance of individual components of our algorithm which highlighted the importance of HER and demonstrations. We think these preliminary results hold promise that *versatile* cloth manipulation by robots is within reach of current machine learning techniques. In the future, we plan to extend our work to real robots by training image to action policies in simulation and transferring the learned policy with domain randomization.

REFERENCES

- [1] S. Miller, J. Van Den Berg, M. Fritz, T. Darrell, K. Goldberg, and P. Abbeel, "A geometric approach to robotic laundry folding," *The International Journal of Robotics Research*, vol. 31, no. 2, pp. 249–267, 2012.
- [2] D. Seita, N. Jamali, M. Laskey, R. Berenstein, A. K. Tanwani, P. Baskaran, S. Iba, J. Canny, and K. Goldberg, "Deep Transfer Learning of Pick Points on Fabric for Robot Bed-Making," in *International Symposium on Robotics Research (ISRR)*, 2019.
- [3] Y. Gao, H. J. Chang, and Y. Demiris, "Iterative path optimisation for personalised dressing assistance using vision and force information," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2016, pp. 4398–4403.
- [4] G. Canal, E. Pignat, G. Alenyà, S. Calinon, and C. Torras, "Joining high-level symbolic planning with low-level motion primitives in adaptive hri: Application to dressing assistance," in *IEEE International Conference on Robotics and Automation*, 2018, pp. 3273–3278.
- [5] Y. Li, Y. Yue, D. Xu, E. Grinspun, and P. K. Allen, "Folding deformable objects using predictive simulation and trajectory optimization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015, pp. 6000–6006.
- [6] M. Cusumano-Towner, A. Singh, S. Miller, J. F. O'Brien, and P. Abbeel, "Bringing clothing into desired configurations with limited perception," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 3893–3900.
- [7] P. Jiménez, "Survey on model-based manipulation planning of deformable objects," *Robotics and computer-integrated manufacturing*, vol. 28, no. 2, pp. 154–163, 2012.
- [8] A. Colomé and C. Torras, "Dimensionality reduction for dynamic movement primitives and application to bimanual manipulation of clothes," *IEEE Transactions on Robotics*, vol. 34, no. 3, pp. 602–615, 2018.
- [9] A. Colomé, A. Planells, and C. Torras, "A friction-model-based framework for reinforcement learning of robotic tasks in non-rigid environments," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 5649–5654.
- [10] F. Ruggiero, V. Lippiello, and B. Siciliano, "Nonprehensile dynamic manipulation: A survey," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1711–1718, 2018.
- [11] Y. Yamakawa, A. Namiki, and M. Ishikawa, "Motion planning for dynamic folding of a cloth with two high-speed robot hands and two high-speed sliders," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 5486–5491.
- [12] J. Maitin-Shepard, M. Cusumano-Towner, J. Lei, and P. Abbeel, "Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 2308–2315.
- [13] F. Osawa, H. Seki, and Y. Kamiya, "Unfolding of massive laundry and classification types by dual manipulator," *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol. 11, no. 5, pp. 457–463, 2007.
- [14] C. Bersch, B. Pitzer, and S. Kammel, "Bimanual robotic cloth manipulation for laundry folding," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 1413–1419.
- [15] E. Corona, G. Alenyà, T. Gabas, and C. Torras, "Active garment recognition and target grasping point detection using deep learning," *Pattern Recognition*, vol. 74, pp. 629–641, 2018.
- [16] A. X. Lee, A. Gupta, H. Lu, S. Levine, and P. Abbeel, "Learning from multiple demonstrations using trajectory-aware non-rigid registration with applications to deformable object manipulation," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 5265–5272.
- [17] B. Balaguer and S. Carpin, "Combining imitation and reinforcement learning to fold deformable planar objects," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 1405–1412.
- [18] J. Matas, S. James, and A. J. Davison, "Sim-to-real reinforcement learning for deformable object manipulation," *arXiv preprint arXiv:1806.07851*, 2018.
- [19] Z. Hu, P. Sun, and J. Pan, "Three-dimensional deformable object manipulation using fast online gaussian process regression," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 979–986, 2018.
- [20] A. Ramisa, G. Alenyà, F. Moreno-Noguer, and C. Torras, "Finddd: A fast 3d descriptor to characterize textiles for robot manipulation," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 824–830.
- [21] K. Kawaharazuka, T. Ogawa, J. Tamura, and C. Nabeshima, "Dynamic manipulation of flexible objects with torque sequence using a deep neural network," *arXiv preprint arXiv:1901.10142*, 2019.
- [22] P.-C. Yang, K. Sasaki, K. Suzuki, K. Kase, S. Sugano, and T. Ogata, "Repeatable folding task by humanoid robot worker using deep learning," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 397–403, 2016.
- [23] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," *GitHub repository*, 2016.
- [24] J. Lee, M. X. Grey, S. Ha, T. Kunz, S. Jain, Y. Ye, S. S. Srinivasa, M. Stilman, and C. K. Liu, "Dart: Dynamic animation and robotics toolkit," *J. Open Source Software*, vol. 3, no. 22, p. 500, 2018.
- [25] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.
- [26] D. Quillen, E. Jang, O. Nachum, C. Finn, J. Ibarz, and S. Levine, "Deep reinforcement learning for vision-based robotic grasping: A simulated comparative evaluation of off-policy methods," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6284–6291.
- [27] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6292–6299.
- [28] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 3389–3396.
- [29] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural networks*, vol. 21, no. 4, pp. 682–697, 2008.
- [30] A. X. Lee, H. Lu, A. Gupta, S. Levine, and P. Abbeel, "Learning force-based manipulation of deformable objects from multiple demonstrations," in *IEEE International Conference on Robotics and Automation*, 2015.
- [31] F. Faure, C. Duriez, H. Delingette, J. Allard, B. Gilles, S. Marchesseau, H. Talbot, H. Courtecuisse, G. Bousquet, I. Peterlik, and S. Cotin, "SOFA: A Multi-Model Framework for Interactive Physical Simulation," in *Soft Tissue Biomechanical Modeling for Computer Assisted Surgery*, ser. Studies in Mechanobiology, Tissue Engineering and Biomaterials, Y. Payan, Ed. Springer, Jun. 2012, vol. 11, pp. 283–321. [Online]. Available: <https://hal.inria.fr/hal-00681539>
- [32] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.
- [33] T. Schaul, D. Horgan, K. Gregor, and D. Silver, "Universal value function approximators," in *International Conference on Machine Learning*, 2015, pp. 1312–1320.
- [34] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. P. Abbeel, and W. Zaremba, "Hindsight experience replay," in *Advances in Neural Information Processing Systems*, 2017, pp. 5048–5058.
- [35] M. Večerík, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," *arXiv preprint arXiv:1707.08817*, 2017.
- [36] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.